(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平11－112990

(43)公開日　平成11年(1999)４月23日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｈ０４Ｎ　7/32		Ｈ０４Ｎ　7/137	Ｚ

審査請求　未請求　請求項の数２　ＯＬ　（全 ９ 頁）

(21)出願番号　特願平9－272416

(22)出願日　平成９年(1997)10月６日

(71)出願人　000006013
三菱電機株式会社
東京都千代田区丸の内二丁目２番３号

(72)発明者　森　正志
東京都千代田区丸の内二丁目２番３号　三菱電機株式会社内

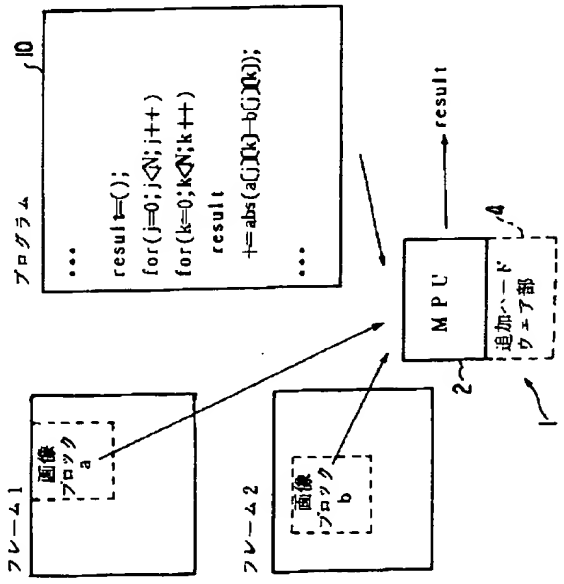
(74)代理人　弁理士　吉田　研二　（外２名）

(54)【発明の名称】　画像圧縮処理装置

(57)【要約】

【課題】　リアルタイムの処理を行うことができると共に、画像圧縮処理アルゴリズムの変更に対して迅速に対応することを目的とする。

【解決手段】　画像圧縮処理装置１は、ＭＰＵ２と追加ハードウェア部４と命令格納部とを有する。命令格納部には、動き補償処理を指示する動き補償処理命令と、画像圧縮処理のうち上記動き補償処理以外の処理を指示する一般処理命令と、を含む画像圧縮処理命令が格納されている。ＭＰＵ２は、一般処理命令を解読及び実行する。追加ハードウェア部４は、動き補償処理命令解読部と動き補償処理命令実行部とブロックレジスタとを有する。動き補償処理命令解読部は、動き補償処理命令を解読する。ブロックレジスタは、画像ブロックを格納する。動き補償処理命令実行部は、解読された動き補償処理命令を、画像ブロックの各画素に対して実行する。



1

【特許請求の範囲】

【請求項１】　複数の画素からなる画像ブロックに対して、動き補償処理を含む画像圧縮処理を行う画像圧縮処理装置において、

前記動き補償処理を指示する動き補償処理命令と、前記画像圧縮処理のうち前記動き補償処理以外の処理を指示する一般処理命令と、を含む画像圧縮処理命令が格納された命令格納手段と、

前記一般処理命令を解読及び実行するＭＰＵと、

前記動き補償処理命令を解読及び実行する追加ハードウェア手段と、を有することを特徴とする画像圧縮処理装置。

【請求項２】　前記画像圧縮処理装置は、画像ブロックが格納されたフレームメモリを有し、

前記追加ハードウェア手段は、前記動き補償処理命令を解読する動き補償処理命令解読手段と、前記フレームメモリ内の画像ブロックを必要に応じて格納するブロックレジスタと、前記動き補償処理命令を、画像ブロックの各画素に対して、実行する動き補償処理命令実行手段と、を有することを特徴とする請求項１に記載の画像圧縮処理装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、画像圧縮処理装置、特に動き補償処理を含む画像圧縮処理を行う画像圧縮処理装置に関する。

【０００２】

【従来の技術】撮像等により得られる画像情報は大きな情報量を有しており、その伝送や管理が容易でないため、通信等に関しては、画像圧縮処理が行われる。

【０００３】画像圧縮処理として、一般的に使用されている動画圧縮方式には、例えばＨシリーズやＭＰＥＧ等がある。これらの標準的な画像圧縮処理において、動画像を効率良く圧縮するための方法として動き補償処理がある。この動き補償処理は、一般的に、その処理負荷が大きい。このため、動き補償処理を含む画像圧縮処理を行う場合において、テレビ会議等の双方向通信、放送分野等のリアルタイム処理の必要があるアプリケーション分野では、専用ハードウェアを使用して画像圧縮処理を行っている。

【０００４】しかしながら、上述の従来技術には、以下に示す問題点がある。この画像圧縮処理技術においては、専用ハードウェアは画像圧縮アルゴリズムに専用に作成されているため、対応する画像圧縮アルゴリズムのみにしか使用できないという問題点がある。このため、画像圧縮アルゴリズムに種々の改良や変更が生じると、ハードウェアを作成し直す必要がある。この結果、画像アルゴリズムの変更等に対して柔軟で迅速に対応することができなかった。

【０００５】そこで、汎用のＭＰＵを使用した画像圧縮処理技術がある。この技術においては、画像圧縮アルゴリズムの改良や変更に対してソフトウェアを置き換えることにより、どのような画像圧縮アルゴリズムに対しても、画像圧縮処理が可能である。この技術は、主に、蓄積系のデータ生成（ＣＤ－ＲＯＭ，ＶＯＤ）等の分野で使用されている。

2

【０００６】

【発明が解決しようとする課題】しかしながら、汎用のＭＰＵを使用した画像圧縮処理技術においては、画像圧縮処理を全て汎用ＭＰＵで処理しているため、動き補償処理のような負荷の重い処理に対しては、リアルタイムの画像圧縮処理を行うことができなかった。

【０００７】本発明は、以上のような問題点を解決するためになされたものであり、その目的は、動き補償処理を含む画像圧縮処理に対して、リアルタイムの処理を行うことができると共に、画像圧縮処理アルゴリズムの変更に対して迅速に対応することができる画像圧縮処理装置を提供することにある。

【０００８】

【課題を解決するための手段】以上のような目的を達成するために、第１の発明に係る画像圧縮処理装置は、複数の画素からなる画像ブロックに対して、動き補償処理を含む画像圧縮処理を行う画像圧縮処理装置において、前記動き補償処理を指示する動き補償処理命令と、前記画像圧縮処理のうち前記動き補償処理以外の処理を指示する一般処理命令と、を含む画像圧縮処理命令が格納された命令格納手段と、前記一般処理命令を解読及び実行するＭＰＵと、前記動き補償処理命令を解読する動き補償処理命令解読手段と、解読された動き補償処理命令を、画像ブロックの各画素に対して、実行する動き補償処理命令実行手段と、を有するものである。

【０００９】第２の発明に係る画像圧縮処理装置は、第１の発明において、前記画像圧縮処理装置は、画像ブロックが格納されたフレームメモリを有し、前記追加ハードウェア手段は、前記動き補償処理命令を解読する動き補償処理命令解読手段と、前記フレームメモリ内の画像ブロックを必要に応じて格納するブロックレジスタと、前記動き補償処理命令を、画像ブロックの各画素に対して、実行する動き補償処理命令実行手段と、を有するものである。

【００１０】

【発明の実施の形態】以下、図面に基づいて本発明の好適な実施の形態について説明する。図１は、本発明の実施の形態である画像圧縮処理装置を示す模式図である。画像圧縮処理装置１は、ＭＰＵ２と追加ハードウェア部４と命令格納部（図示せず）とを有する。

【００１１】命令格納部には、動き補償処理を指示する動き補償処理命令と、画像圧縮処理のうち上記動き補償処理以外の処理を指示する一般処理命令と、を含む画像圧縮処理命令が格納されている。この画像圧縮処理命令

は、例えばＣ言語で書かれたプログラムが機械語等に変換されて、命令格納部に格納される。なお、上記プログラムはＣ言語以外で書かれていてもよい。

【００１２】ＭＰＵ２は、一般処理命令を解読及び実行する。ＭＰＵ２は、複数の常識的な算術演算が可能な３２ｂｉｔレジスタ（以下、汎用レジスタ）を複数個有する。また、ＭＰＵ２は、レジスタ値アドレッシングや直接値アドレッシング等のアドレッシングモードが可能であると仮定する。

【００１３】追加ハードウェア部４は、動き補償処理命令解読部と、動き補償処理命令実行部と、ブロックレジスタと、を有する。

【００１４】ブロックレジスタは、フレームメモリに格納された画像ブロックが必要に応じて格納される。動き補償処理命令解読部は、動き補償処理命令を解読する。動き補償処理命令実行部は、解読された動き補償処理命令を、画像ブロックの各画素に対して実行する。なお、追加ハードウェア部４には、専用レジスタが設けられている。

【００１５】このように構成された画像圧縮処理装置１においては、先ず、プログラム１０が命令格納部にロードされて、機械語に変換された各命令が格納される。そして、一般命令はＭＰＵ２で解読、実行される。動き補償処理命令は追加ハードウェア部４で解読、実行される。この際、ブロックレジスタには、フレーム１、２の画像ブロックａ，ｂが格納される。そして、動き補償処理命令実行部により、上記画像ブロックａ，ｂの各画素に対して、動き補償処理を実行する。これにより、動き補償に関する情報resultが出力される。

【００１６】本実施の形態においては、画像圧縮処理を指示する画像圧縮処理命令を、処理負荷の重さにより、動き補償処理命令と、それ以外の一般処理命令と、に分けて命令格納部に格納している。そして、処理負荷の重い動き補償処理命令は、動き補償命令解読部及び動き補償命令実行部により、解読及び実行している。即ち、これらの動き補償命令解読部及び動き補償命令実行部が専用ハードウェアとなる。このため、動き補償処理を含む画像圧縮処理についてもリアルタイム処理が可能となる。

【００１７】一方、処理負荷の軽い一般処理命令は、汎用のＭＰＵ２により解読及び実行している。このため、従来技術のように画像圧縮アルゴリズムを達成するためのハードウェア構成を全てそのアルゴリズムに専用のものにしないで、動き補償命令解読部及び動き補償命令実行部のみを専用のハードウェアとしている。これにより、ソフトウェアを変更して、動き補償処理命令と一般処理命令とを適切に組み合わせ直すことにより、画像圧縮アルゴリズムの変更に対しても、柔軟で迅速に対応することができる。

【００１８】なお、本実施の形態においては、動き補償処理に必要な命令は、後述する追加命令セットに分類、整理される。追加命令セットとは、追加ハードウェア部４に対する命令である。この際、追加命令セットに必要となる追加レジスタセットも分類、整理される。追加レジスタセットとは、専用レジスタ及びブロックレジスタに対する命令である。以下、追加レジスタセット、追加命令セットについて説明する。

【００１９】（追加レジスタセット及び追加命令セットの説明）なお、以下で示す型char,short,intはそれぞれ符号付きの８ビット、１６ビット、３２ビットの整数とする。そして、これらの型char,short,intにそれぞれunsignedが付された場合、その同一の長さの符号がない整数を意味するものとする。

【００２０】先ず、追加レジスタセットについて説明する。なお、汎用レジスタは、Ｒｘ（ｘ＝０．．ｍ－１）と記載する。

【００２１】［１］追加レジスタセット

（１）専用レジスタセット

専用レジスタを指定する場合、プログラム上でＣＲｘ（ｘ＝０．．ｍ－１）と記載する。そして、このＣＲｘを利用した命令として以下のものがある。

【００２２】（ａ）ＭＯＤＥ（ＣＲ０）

これは、ブロックレジスタの関連命令が同期的（０）／非同期的（１）に動作するか否かを示す。型は、unsigned intである。

【００２３】（ｂ）ＳＴＡＴ（ＣＲ１）

これは非同期モードで追加部分の動作（０）／動作中（１）を示す。型はunsigned intである。

【００２４】（ｃ）ＰＩＸ_ＳＩＺＥ（ＣＲ２）

これは各画素のバイト単位での大きさ（１、２、４）を示す。この型はunsigned intである。

【００２５】（ｄ）ＦＲＡＭＥ_ＷＩＤＦＨ（ＣＲ３）

ブロックの辺（正方形）の長さ（画素数）を示す。この型はunsigned intである。

【００２６】（ｅ）ＢＬＯＣＫ_ＳＩＺＥ（ＣＲ４）

これはフレームメモリの水平方向の長さを示す。この型はunsigned intである。

【００２７】（２）ブロックレジスタセット

ブロックレジスタを指定する場合、プログラム上でＢＬＯＣＫｘ（ｘ＝０．．ｎ－１）と記載する。これはフレーム内のメモリを格納することを示し、型はint(１６＊１６）である。

【００２８】次に、追加命令セットについて説明する。

【００２９】［２］追加命令セット

以下において演算はＣ言語的に記載する。また、以下でＰＩＸ_ＴＹＰＥ（ＰＩＸ_ＳＩＺＥ)の型は以下の通りである。

【００３０】ｃｈａｒ　（ＰＩＸ_ＴＹＰＥが１のとき）

ｓｈｏｒｔ（ＰＩＸ_ＴＹＰＥが１のとき）

int　（PIX_TYPEが1のとき）

（1）専用レジスタロード／ストア命令

（a）LD CRx，Ry （x=0．．．4，y=0．．．m-1）

これは汎用レジスタに格納されたデータ等を専用レジスタに代入する命令である。この命令は、CRx=RyでC言語プログラム上に示される。

【0031】（b）LD Ry，CRx （x=0．．．4，y=0．．．m-1）

この命令は専用レジスタに格納されたデータ等を汎用レジスタに代入するものである。これは、Ry=CRxでC言語プログラム上に示される。

【0032】（2）ブロックレジスタロード／ストア命令

（a）LD BLOCKx，（addr）

これはブロックレジスタにアドレスaddrからBLOCK_SIZE*BLOCK_SIZE画素分のデータを代入する命令である。これをC言語を用いた演算で示すと、図2（a）に示すプログラムリストで示される。

【0033】（b）LD BLOCKx，（Rx）

これはブロックレジスタにRxで指定されたアドレスからBLOCK_SIZE*BLOCK_SIZE画素分のデータを代入する命令である。これをC言語を用いた演算で示すと、図2（b）に示すプログラムリストで示される。

【0034】（c）ST （addr），BLOCKx

これはブロックレジスタからアドレス addrへBLOCK_SIZE*BLOCK_SIZE画素分のデータを代入する命令である。これをC言語を用いた演算で示すと、図3（a）に示すプログラムリストで示される。

【0035】（d）ST （Rx），BLOCKx

これはブロックレジスタからアドレス addrへBLOCK_SIZE*BLOCK_SIZE画素分のデータを代入する命令である。これをC言語を用いた演算で示すと、図3（b）に示すプログラムリストで示される。

【0036】（e）PACK BLOCKx，（addr）

これはブロックレジスタにフレームメモリ内のアドレスaddrから辺の長さがBLOCK_SIZE（画素）の画像ブロックのデータを代入する命令である。これをC言語を用いた演算で示すと、図4（a）に示すプログラムリストで示される。

【0037】（f）PACK BLOCKx，（Ry）

これはブロックレジスタにフレームメモリ内のアドレスRyから辺の長さがBLOCK_SIZE（画素）の画像ブロックのデータを代入する命令である。これをC言語を用いた演算で示すと、図4（b）に示すプログラムリストで示される。

【0038】（g）UNPACK （addr），BLOCKx

これはブロックレジスタからフレームメモリ内のアドレスaddrに辺の長さがBLOCK_SIZE（画素）の画像ブロックのデータを代入する命令である。これをC言語を用いた演算で示すと、図5（a）に示すプログラムリストで示される。

【0039】（h）UNPACK （Ry），BLOCKx

これはブロックレジスタからフレームメモリ内のアドレスRyに辺の長さがBLOCK_SIZE（画素）の画像ブロックのデータを代入する命令である。これをC言語を用いた演算で示すと、図5（b）に示すプログラムリストで示される。

【0040】（3）ブロックレジスタ演算処理命令セット

先ず、以下に示すプログラムリストにおいて示されるPIX_TYPE（PIX_SIZE)の型、MAX，MINの定義について示す。

【0041】PIX_TYPE（PIX_SIZE)の型は以下の通りである。

【0042】char （PIX_TYPEが1のとき）

short （PIX_TYPEが2のとき）

int （PIX_TYPEが4のとき）

また、MAX，MINは、

（MIN、MAX）=（-128、127）

∴（PIX_TYPE=1のとき）

（MIN、MAX）=（-32768、32767）

∴（PIX_TYPE=2のとき）

（MIN、MAX）=（-2147483648、2147483647）

∴（PIX_TYPE=4のとき）

として、次のようにCLIP（x），UCLIP（x）を定義する。

【0043】#define CLIP（x）：

（（（x）>MAX）?MAX：（（（x>MAX）?MAX：（（（X）<MIN）?MIN：（x））））

#define UCLIP（x）：（（（x）>MAX）?MAX：（x））

次に、各ブロックレジスタ演算命令セットを以下に示す。

【0044】（a）DIFF BLOCKx，BLOCKy，BLOCKz

これは画像ブロック間の差分を計算する命令である。これをC言語を用いた演算で示すと、図6（a）に示すプログラムリストで示される。

【0045】（b）SQ BLOCKx，BLOCKy

これは画像ブロックの各画素の2乗を計算する命令である。これをC言語を用いた演算で示すと、図6（b）に示すプログラムリストで示される。

【0046】（c）ABS BLOCKx，BLOCK

y
これは画像ブロックの各画素の絶対値を計算する命令である。これをＣ言語を用いた演算で示すと、図７（ａ）に示すプログラムリストで示される。
【００４７】（ｄ）ＳＵＭ　Ｒｘ，ＢＬＯＣＫｙ
これは画像ブロック内の各画素の総和を計算する命令である。これをＣ言語を用いた演算で示すと、図７（ｂ）に示すプログラムリストで示される。
【００４８】（ｅ）ＡＤＤＫ　ＢＬＯＣＫｘ　ＢＬＯＣＫｙ　Ｒｚ
これは画像ブロック内の各画素に定数値を加える命令である。これをＣ言語を用いた演算で示すと、図８（ａ）に示すプログラムリストで示される。
【００４９】なお、高速化のためには適切なフレームメモリに関するキャシング機構が必要である。ブロックレジスタロード／ストア命令に関するアクセス順序は通常ラスタスキャンの順番であると仮定できる。このため、次にどのようなデータをブロックレジスタにＬＯＡＤ，ＰＡＣＫするかはある程度見当がつく（例えばブロックのちょうど右側の部分や下の部分）。但し、ＰＩＸ_ＳＩＺＥ，ＦＲＡＭＥ_ＷＩＤＴＨ，ＢＬＯＣＫ_ＳＩＺＥに変更があれば、キャッシュはフラッシュされる必要がある。
【００５０】以上のような各追加レジスタセット及び各追加命令セットを用いることにより、動き補償処理を分類及び整理できる。そして、これらの追加レジスタセット及び追加命令セットを用いた画像圧縮処理プログラムを、命令格納部に格納して追加ハードウェア部４で解読、実行することにより、動き補償処理を少ない命令数で実行することが可能となる。
【００５１】そして、上述の画像圧縮処理装置と、上述の追加レジスタセット及び追加命令セットを使用した画像圧縮処理プログラムと、を利用した動き補償処理例について以下に示す。
【００５２】（動き補償処理例）Ｈ．２６１やＭＰＥＧ等における動き補償処理では、画像ブロックに対して符号化しようとして何らかの検索アルゴリズムで何らかの評価関数が（近似的に）最小になるように前のフレームで検索処理を行うものである。この際、検索アルゴリズムは、現画像ブロックの位置に対応する前フレーム内での前画像ブロックの近傍の全画素数検索とし、評価関数は絶対差分和とする。なお、本方式ではより複雑なサーチアルゴリズム、評価関数を用いることも可能である。図９は、上記検索アルゴリズムを適用したＣ言語プログラムのリストを示す図であり、図１０は、動き補償処理に使用する現在フレーム及び１つ前のフレームを示す図である。図９に示すプログラムで用いられている変数の定義を以下に示す。
【００５３】prev_frame：１つ前のフレームのアドレス
curr_frame：現在のフレームのアドレス
x,y：現在のブロックの左角の位置
x_min,y_min,x_max,y_max：前のフレームでのサーチ範囲の上限、下限のｘ、ｙ座標
R0,R1,R2：ＭＰＵの汎用レジスタで、今のところＣ言語は使用していない。
【００５４】width：フレームの幅
size：画像ブロックの幅
pixel_size：画素の大きさ
e：画像ブロック間の差分値
上述の処理により、図９に示す位置x_pos,y_posに、それぞれ、所定範囲内での絶対差分和値が最小となる位置が算出される。
【００５５】本実施の形態においては、最も時間のかかる動き補償処理の高速化を図ることができる。そして、その他の処理については、ＭＰＵの一般処理命令で実行するため、種々の画像圧縮処理アルゴリズムや画像圧縮処理アルゴリズムの変更に対しても、柔軟で迅速に対応可能となる。
【００５６】
【発明の効果】請求項１に記載の発明によれば、処理負荷の重い動き補償処理命令は、専用のハードウェアとしての追加ハードウェア手段により、解読及び実行している。このため、動き補償処理を含む画像圧縮処理についてもリアルタイム処理が可能となる。一方、処理負荷の軽い一般処理命令は、汎用のＭＰＵにより解読及び実行している。これにより、ソフトウェアを変更して、動き補償処理命令と一般処理命令とを適切に組み合わせ直すことにより、画像圧縮アルゴリズムの変更に対しても、柔軟で迅速に対応することができる。
【００５７】また、請求項２に記載の発明によれば、追加ハードウェア手段は、動き補償処理解読手段、動き補償処理実行手段及びブロックレジスタに分けることができる。これにより、画像圧縮アルゴリズムにおいて、例えば動き補償処理解読に関する部分のみの変更の場合、追加ハードウェア手段に関する部分を全部変更するのではなく、補償処理解読に関する部分だけを変更すればよい。
【図面の簡単な説明】
【図１】　本発明の実施の形態である画像圧縮処理装置を示す模式図である。
【図２】　（ａ），（ｂ）は追加命令セットのうち、ブロックレジスタロード／ストア命令を示すプログラムリストを示す図である。
【図３】　同じく、（ａ），（ｂ）は追加命令セットのうち、ブロックレジスタロード／ストア命令を示すプログラムリストを示す図である。
【図４】　同じく、（ａ），（ｂ）は追加命令セットのうち、ブロックレジスタロード／ストア命令を示すプログラムリストを示す図である。
【図５】　同じく、（ａ），（ｂ）は追加命令セットの

うち、ブロックレジスタロード／ストア命令を示すプログラムリストを示す図である。

【図６】　（ａ）は、ブロックレジスタ演算命令セットのうち、画像ブロック間の差分を計算する命令を示すプログラムリストを示す図であり、（ｂ）は同じくブロックレジスタ演算命令セットのうち、画像ブロックの各画素の２乗を計算する命令を示すプログラムリストを示す図である。

【図７】　（ａ）は、ブロックレジスタ演算命令セットのうち、画像ブロック間の各画素の絶対値を計算する命令を示すプログラムリストを示す図であり、（ｂ）は同じくブロックレジスタ演算命令セットのうち、画像ブロック内の各画素の総和を計算する命令を示すプログラムリストを示す図である。

【図８】　ブロックレジスタ演算命令セットのうち、画像ブロック内の各画素に定数値を加える命令を示すプログラムリストを示す図である。

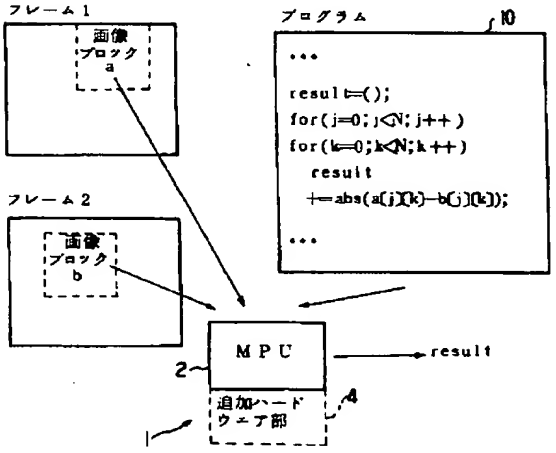
【図９】　動き補償処理を行うためのプログラムリストを示す図である。

【図１０】　動き補償処理に使用する現在フレーム及び１つ前のフレームを示す図である。

【符号の説明】

１　画像圧縮処理装置、２　ＭＰＵ、４　追加ハードウェア部、１０　プログラム。

【図１】



【図２】

（ａ）ＬＤ　ＢＬＯＣＫｘ（ａｄｄｒ）のプログラムリスト

```
int i;
PIX_TYPE(PIX_SIZE) *dst, *src;

dst = (PIX_TYPE(PIX_SIZE) *)BLOCKx;
src = (PIX_TYPE(PIX_SIZE) *)addr;

for (i = 0; i < BLOCK_SIZE * BLOCK_SIZE; i++)
        *dst++ = *src++;
```

（ｂ）ＬＤ　ＢＬＯＣＫｘ（Ｒｘ）のプログラムリスト

```
int i;
PIX_TYPE(PIX_SIZE) *dst, *src;

dst = (PIX_TYPE(PIX_SIZE) *)BLOCKx;
src = (PIX_TYPE(PIX_SIZE) *)Rx;

for (i = 0; i < BLOCK_SIZE * BLOCK_SIZE; i++)
        *dst++ = *src++;
```

【図８】

ＡＤＤＫ　ＢＬＯＣＫｘ，ＢＬＯＣＫｙ，Ｒｚのプログラムリスト

```
int i;
PIX_TYPE(PIX_SIZE) *dst, *src;

src = (PIX_TYPE(PIX_SIZE) *)BLOCKy;
dst = (PIX_TYPE(PIX_SIZE) *)BLOCKx;

for (i = 0; i < BLOCK_SIZE * BLOCK_SIZE; i++){
        int temp;

        temp = *src++;
        temp += Rz;
        *dst++ = CLIP(temp);
}
```

【図3】

(a) ST (addr) BLOCKxのプログラムリスト

```
int i;
PIX_TYPE(PIX_SIZE) *dst, *src;

src = (PIX_TYPE(PIX_SIZE) *)BLOCKx;
dst = (PIX_TYPE(PIX_SIZE) *)addr;

for (i = 0; i < BLOCK_SIZE * BLOCK_SIZE; i++)
        *dst++ = *src++;
```

(b) ST (Rx) BLOCKxのプログラムリスト

```
int i;
PIX_TYPE(PIX_SIZE) *dst, *src;

src = (PIX_TYPE(PIX_SIZE) *)BLOCKx;
dst = (PIX_TYPE(PIX_SIZE) *)Rx;

for (i = 0; i < BLOCK_SIZE * BLOCK_SIZE; i++)
        *dst++ = *src++;
```

【図4】

(a) PACK BLOCKx (addr) のプログラムリスト

```
int i, j;
PIX_TYPE(PIX_SIZE) *dst, *src;

dst = (PIX_TYPE(PIX_SIZE) *)BLOCKx;
src = (PIX_TYPE(PIX_SIZE) *)addr;

for (i = 0; i < BLOCK_SIZE; i++){
        for (j = 0; j < BLOCK_SIZE; j++)
                *dst++ = *src++;
        src += (FRAME_WIDTH - BLOCK_SIZE);
}
```

(b) PACK BLOCKx (Ry) のプログラムリスト

```
int i, j;
PIX_TYPE(PIX_SIZE) *dst, *src;

dst = (PIX_TYPE(PIX_SIZE) *)BLOCKx;
src = (PIX_TYPE(PIX_SIZE) *)Ry;

for (i = 0; i < BLOCK_SIZE; i++){
        for (j = 0; j < BLOCK_SIZE; j++)
                *dst++ = *src++;
        src += (FRAME_WIDTH - BLOCK_SIZE);
}
```

【図7】

(a) ABS BLOCKx, BLOCKyのプログラムリスト

```
int i;
PIX_TYPE(PIX_SIZE) *dst, *src;

src = (PIX_TYPE(PIX_SIZE) *)BLOCKy;
dst = (PIX_TYPE(PIX_SIZE) *)BLOCKx;

for (i = 0; i < BLOCK_SIZE * BLOCK_SIZE; i++){
        int temp;

        temp = *src++;
        temp = abs(temp);
        *dst++ = CLIP(temp);
}
```

(b) SUM Rx, BLOCKyのプログラムリスト

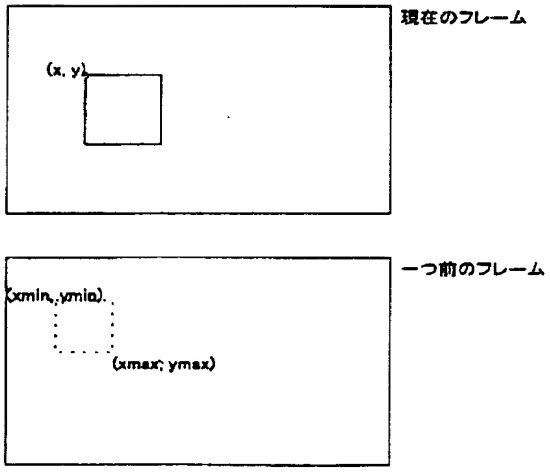
```
int i;
PIX_TYPE(PIX_SIZE) *dst, *src;

src = (PIX_TYPE(PIX_SIZE) *)BLOCKy;

Rx = 0;
for (i = 0; i < BLOCK_SIZE * BLOCK_SIZE; i++){
        int temp;

        Rx += *src++;
}
```

【図10】



【図5】

(a) UNPACK (addr) BLOCKxのプログラムリスト

```
int i, j;
PIX_TYPE(PIX_SIZE) *dst, *src;

dst = (PIX_TYPE(PIX_SIZE) *)addr;
src = (PIX_TYPE(PIX_SIZE) *)BLOCKx;

for (i = 0; i < BLOCK_SIZE; i++){
        for (j = 0; j < BLOCK_SIZE; j++)
                *dst++ = *src++;
        dst += (FRAME_WIDTH - BLOCK_SIZE);
}
```

(b) UNPACK (Ry) BLOCKxのプログラムリスト

```
int i, j;
PIX_TYPE(PIX_SIZE) *dst, *src;

dst = (PIX_TYPE(PIX_SIZE) *)Ry;
src = (PIX_TYPE(PIX_SIZE) *)BLOCKx;

for (i = 0; i < BLOCK_SIZE; i++){
        for (j = 0; j < BLOCK_SIZE; j++)
                *dst++ = *src++;
        dst += (FRAME_WIDTH - BLOCK_SIZE);
}
```

【図6】

(a) DIFF BLOCKx, BLOCKy, BLOCKzのプログラムリスト

```
int i;
PIX_TYPE(PIX_SIZE) *dst, *src0, *src1;

src0 = (PIX_TYPE(PIX_SIZE) *)BLOCKy;
src1 = (PIX_TYPE(PIX_SIZE) *)BLOCKz;
dst = (PIX_TYPE(PIX_SIZE) *)BLOCKx;

for (i = 0; i < BLOCK_SIZE * BLOCK_SIZE; i++){
        int temp;

        temp = *src0++;
        temp -= *src1++;
        *dst++ = CLIP(temp);
}
```

(b) SQ BLOCKx, BLOCKyのプログラムリスト

```
int i;
PIX_TYPE(PIX_SIZE) *dst, *src;

src = (PIX_TYPE(PIX_SIZE) *)BLOCKy;
dst = (PIX_TYPE(PIX_SIZE) *)BLOCKx;

for (i = 0; i < BLOCK_SIZE * BLOCK_SIZE; i++){
        int temp;

        temp = *src++;
        temp *= temp;
        *dst++ = CLIP(temp);
}
```

【図９】

```
int i, j, e;

e = MAX;
y_pos = y_min;
x_pos = x_min;

R1 = curr_frame + width * y + x;

LD      PIX_SIZE,       [pix_size]
LD      FRAME_WIDTH,    [width]
LD      BLOCK_SIZE,     [size]

PACK    BLOCK0,         [R1]
for (i = y_min; i <= y_max; i++)
for (j = x_min; j <= x_max; j++){
        R0 = prev_frame + width * i + j;

        PACK    BLOCK1,         [R0]
        DIFF    BLOCK2,         BLOCK1,         BLOCK0
        ABS     BLOCK1,         BLOCK2
        SUM     R2,             BLOCK1

        if (R2 < e){
                x_pos = j;
                y_pos = i;
                e = R2;
        }
}
```